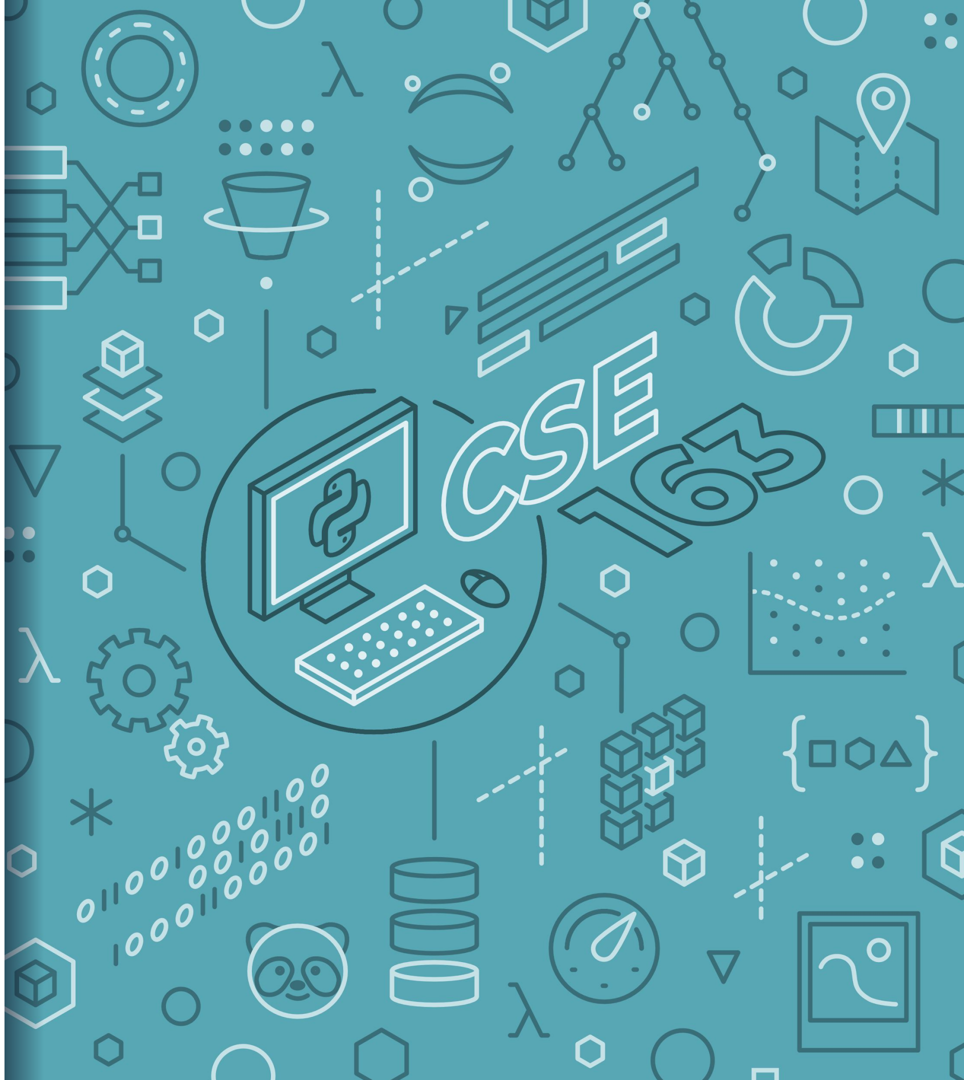
CSE 163

Lists and Sets and Dictionaries!

Oh My!

Hunter Schafer



Last Time Time

- Lists
- Working with Files
- Documenting Code
- None
- Homework Logistics

This Time

- More Lists
- Sets
- Dictionaries
- Tuples?

Lists

- A List is a generic **collection** that holds multiple values
  - Each value has an index
  - String is like a list of length 1 strings
- A list can store multiple values of any types

```
l1 = [1, 2, 3, 4]
l2 = ['hello', 'goodbye']
l3 = [1, 'dog', 3.4]
```

- See List Demo

Strings

vs.

Lists

```
s = 'hello world'
# Length
len(s)  # 11

# Indexing
s[1]            # 'e'
s[len(s) - 1]   # 'd'

# Looping
for i in range(len(s)):
   print(s[i])

for c in s:
    print(c)
```

```
l = ['dog', 'says', 'woof']
# Length
len(l)  # 3

# Indexing
l[1]            # 'says'
l[len(l) - 1]   # 'woof'

# Looping
for i in range(len(l)):
   print(l[i])

for word in l:
    print(word)
```

List methods

- List objects have methods you can call to change their state

`list.append(x)`	Adds x to the end
`list.extend(xs)`	Adds all elements in xs at the end
`list.insert(i, x)`	Inserts x at index i
`list.remove(x)`	Removes the first instance of x
`list.pop([i])`	Removes the value at index i (default: last)
`list.clear()`	Removes all values
`list.index(x)`	Returns the index of the given value
`list.reverse()`	Reverses the elements
`list.sort()`	Sorts the elements

List Demo

- List Demo
  - Build up a list
  - Remove values
  - in keyword
  - Introduction to list comprehensions

in keyword:

```
xs = [1, 2, 3]


if 2 in xs:
    print('Found it!')
```

List Comprehensions

```
squares = [i ** 2 for i in range(1, 11) if i % 2 == 0]
```

```
squares = [
    i ** 2
    for i in range(1, 11)
    if i % 2 == 0
]
```

4) Put it all in a list

3) Expression with loop variable

1) What are you looping over?

2) [Optional] Skip if this is false

Check your understanding!

What is list created by this comprehension?

```
words = ['I', 'saw', 'a', 'dog', 'today']
[word[1] for word in words if len(word) >= 2]
```

Recap: Lists

- Can store more than one value, each one has an index
- Can index into them (brackets) to get or assign values
- Have methods to add/remove values
- Can be created with a list comprehension

Text Data Analysis

- When doing data analysis on text data, it’s useful to have some idea about the data you are processing.
- **First**: How many unique words are there?
- **Later**: What is the most frequent word in the file?

Text Data Analysis Demo

Set

- Data structure highly optimized for membership queries
  - "if x in set" is very fast
- Acts a lot like a list, BUT
  - Does not allow indexing operations
  - Does not allow duplicates

Set Methods

`set()`	Makes an empty set
`set.add(x)`	Adds x to the end
`set.remove(x)`	Removes x from this set
`set.clear()`	Removes all values from this set

[Set Demo]

Brain Break

Poll Everywhere

Think

1 minute

pollev.com/cse163

Suppose we run the following chunk of code, what is the output?

```
def remove_evens(words):
    for i in range(len(words)):
        if len(words[i]) % 2 == 0:
            words.pop(i)

l = ["a", "bc", "de", "f"]
remove_evens(l)
print(l)
```

- [“a”, “f”]
- [“a”, “de”, “f”]
- [“a”, “bc”, “de”, “f”]
- No output, there is some error thrown

Poll Everywhere

Pair

2 minutes

pollev.com/cse163

Suppose we run the following chunk of code, what is the output?

```
def remove_evens(words):
    for i in range(len(words)):
        if len(words[i]) % 2 == 0:
            words.pop(i)


l = ["a", "bc", "de", "f"]
remove_evens(l)
print(l)
```

- [“a”, “f”]
- [“a”, “de”, “f”]
- [“a”, “bc”, “de”, “f”]
- No output, there is some error thrown

Poll Everywhere

“a”	“bc” “de”	“de” “f”	“f”
0	1	2	3

i | 0 1 2 |

IndexError: list index out of range

range(0, 4, 1)

```
def remove_evens(words):
    for i in range(len(words)):
        if len(words[i]) % 2 == 0:
            words.pop(i)

l = ["a", "bc", "de", "f"]
remove_evens(l)
print(l)
```

Using Lists to Count

- When doing data analysis on text data, it’s useful to have some idea about the data you are processing.
- ~~**First**: How many unique words are there?~~
- **Now**: What is the most frequent word in the file?

Basic Idea

- Need some structure to associate the count to each word.
- It would be nice if we could use a list with words for the index

2	14	15	3
“scurvy”	“a”	“whale”	“moby”

- Lists don’t let us do this because the indices must be ints.

Dictionary

- Python has a data structure called a **dictionary** that allows us to use arbitrary data as indices (keys)

```
d = {}
d['a'] = 1
d['b'] = 2
print(d)  # {'a': 1, 'b': 2}
```

- Dictionary Demo + Text Data Analysis Demo 2
  - Keys are unique, values are not
  - Basically a fancy list, most syntax is familiar
- Will start Wed by going more in-depth with dictionaries

Tuples

- The last Python data structure you should know about is called a **tuple**
- A tuple is a lot like a list, but is immutable! (can't change it)
    - Uses parentheses as start/end

```
t = (1, "hello", 3.4)
print(t)                # (1, 'hello', 3.4)
print(t[0])             # 1
t[1] = 'goodbye'        # Error!

# You can "unpack" tuples to get their contents
a, b, c = t
print(b)                # hello
```

Tuples cont.

- Commonly used to return multiple values from a function

```
def first_two(word):
    return  word[0], word[1]

print(first_two('goodbye'))  # ('g', 'o')

# Can unpack here as well
a, b = first_two('goodbye')
```

Recap

- Learned about 4 different data structures today!

1) **List**
   Has indices, this can be added/removed at any place
2) **Set**
   No duplicates, fast membership queries, no order
3) **Dictionary**
   Like a generic list that can have any value as keys
   Keys are unique, values are not
4) **Tuple**
   Immutable list, commonly used as returns, can “unpack”

Next Time

- More on dictionaries
- Nested data structures
- Pandas

Before Next Time

- Keep up with practice!
- Start the assignment!